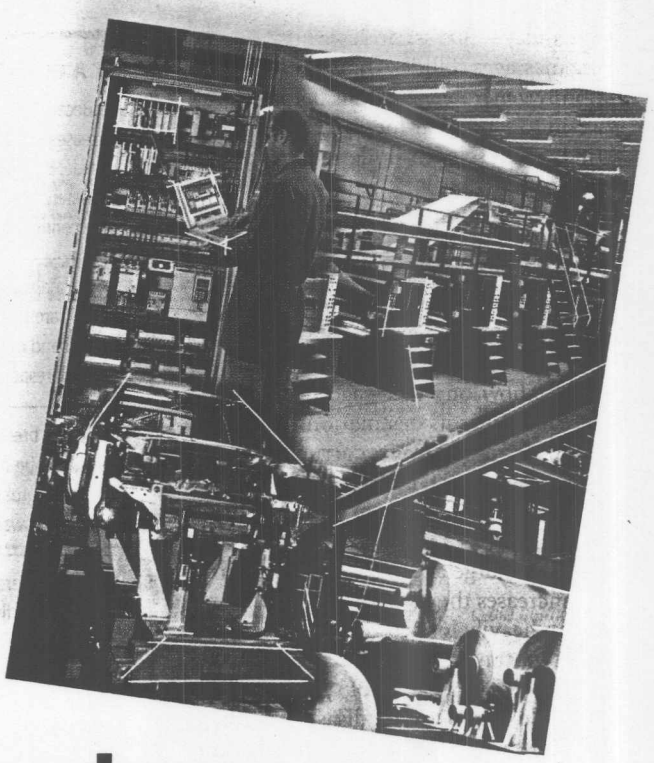
 By Dan Strassberg, Senior Technical Editor



FACTORY-FLOOR COMMUNICATIONS IS FLIRTING WITH WHAT ONCE WERE PURELY OFFICE TECHNOLOGIES. ALTHOUGH WEB PROTOCOLS' MIGRATION FROM E-MAIL TO MUSCULAR MACHINERY ISN'T TOTALLY PROBLEM-FREE, THE WEB'S ROMANCE WITH MANUFACTURING AUGURS WELL FOR A LONGER, HAPPIER RELATIONSHIP THAN MOST IN HIGH TECH.

Web Technology: Sticking its nodes in everybody's business

THE NETWORKING TECHNOLOGY that underlies the World Wide Web is transforming not only the ways in which businesses market and sell their wares but also the ways by which machines communicate with each other. Whereas manufacturers of industrial controls have spent decades constructing a Tower of Babel—in the form of a welter of incompatible, proprietary networking protocols—the ubiquitous Ethernet and TCP are bidding to restore some sanity to industrial networks.

Ethernet and associated protocols began life with the idea that they would connect computers with peripherals and that companies would deploy those devices exclusively within benign and tidy office environments (see **sidebar** "A network-protocol primer"). Today, however, such notions are history. The previously prissy protocols are sticking their nodes into everybody's business. As a result, the Web's networking technology is turning up with increasing regularity in the muscular machinery of the rust belt.

In the days before 10- and 100-Mbps Ethernet (let-alone Gigabit Ethernet), it was common knowledge that Ethernet's physical-layer protocol, CSMA/CD, was unsuited to real-time applications. The protocol specifies that, before a node can transmit data, it must listen to the network and make sure that no other node is transmitting.

Nodes that attempt to transmit simul-

taneously—that is, nodes whose outgoing messages collide—must wait for randomly generated intervals before trying again to send their messages. The idea is that, because each node independently determines how long to wait and chooses its delay at random, the probability is small that, on the next attempt, the nodes' outgoing messages will again collide.

The system works well, but it does not allow you to predict (except statistically) the latency of data. That is, you cannot predict how long a message must wait before it reaches its destination. Moreover, if the message becomes corrupted during transmission, the receiving node detects errors and can request a retransmission, which can cause further unpredictable delays. A large amount of message traffic increases the likelihood of collisions, so on heavily loaded networks, retransmission can cause appreciable delays.

In a real-time control system, unpredictable delays can be disastrous. Imagine the carnage if the valve that controls the flow of hydrofluoric acid into a tank doesn't close until *after* the tank is full, or if the motors that drive the pinch rollers in a mill that forms steel into a 20-ft-wide rolls don't slow down in synchronism!

AT A GLANCE

▷ Because of the dramatic network-speed increases that have taken place in the last decade, Ethernet, despite its nondeterministic latency, is fast enough for many real-time control applications.

▷ Standard Web protocols, such as TCP/IP, are less than ideal for real-time control and can lead to unacceptably slow network response.

▷ A new breed of software, real-time publish-subscribe middleware, can overcome the speed problems of the more familiar network-software protocols.

▷ IC designers have cut the TCP stack's gate count to the point where the necessary hardware is just a tiny part of more complex ICs, thus enabling individual sensors to interface at almost no extra cost to networks based on Web technology.

▷ It makes little sense for individual sensors to publish their own Web pages to company intranets. Reserve that job for higher level devices that aggregate the outputs of multiple sensors.

FAST ENOUGH FOR MANY PURPOSES

Nevertheless, more and more control-system designers recognize that increasing network data rates have invalidated the bygone era's common knowledge. Ethernet is fast enough for many control systems—even in time-critical applications. Ethernet networks that transmit kilohertz data in real time are rather common. Ethernet's suitability depends on the extent of the network, the traffic levels, the error rates, and how critical delays can be. The latency *is* unpredictable, but if no hard failures occur, data rarely fails to reach its destination.

When analysis and measurements show that Ethernet can work acceptably, the logic of using it is often inescapable. Wiring is usually the largest part of the network's cost. If the wiring is already present, the network cost drops dramatically. With Ethernet, the wiring is usually already present. Wherever you go today within the factories of US industry, an Ethernet network is rarely more than a few feet away.

Of course, just because an existing network is close at hand doesn't *automatically* make it the right choice. For example, if the network is already heavily loaded, the additional traffic that the new applications add can become the straw that breaks the camel's back. On the other hand, Ethernet can be a good choice even if you must install a new network. Compared with other architectures, Ethernet hardware and support tools are reasonably priced and readily available, as is expertise in deploying and maintaining Ethernet networks.

ELIMINATING THE SOFTWARE BOTTLENECK

But although the Ethernet physical layer often works well enough to handle real-time control, the protocols that are common in Ethernet-based office applications may not. When a networked application needs to be more responsive, a relatively new type of software, RTPS (real-time publish-subscribe) middleware can come to the rescue (**Reference 1**). Middleware is a software layer that stands between your application and the operating system's networking facilities. Examples of such software are Real-time Innovations' NDDS and National Instruments' Data Socket.

Under the publish-subscribe approach, subscribing nodes specify the type of information they need and how often they require updates. Because the subscribers establish their information needs at the outset, this approach minimizes traffic associated with requests for information. In addition, the RTPS approach can avoid wasting bandwidth on repeated transmissions of nonessential data that a subscriber fails to receive successfully. When a subscriber establishes how often it must receive data, it need not specify exactly which data points it must have. In such cases, if a data point fails to arrive, the next such point may suffice—as long as that point arrives within the subscriber's specified maximum time between updates.

Of course, some subscribers must receive every message of a particular type. Moreover, the messages must arrive error free, and if they do not arrive in the order in which they were sent, the subscriber must be able to place them in that order. (An example of such message and subscriber types is a sequence of commands transmitted to a command processor.) The subscriber can specify these requirements at the outset. By time-stamping every transaction, the RTPS software enables subscribers to place messages in their intended order and to compensate for the unpredictability of delays.

In a happy coincidence, Web technology enables designers of Web-based real-

(continued on page 66)

ACRONYMS

CORBA: Common Object-Request-Broker Architecture
CSMA/CD: carrier-sense multiple access with collision detection
DCOM: Distributed Component Object Model
DNS: domain-name service
HTML: Hypertext Markup Language
HTTP: Hypertext Transfer Protocol
ICMP: Internet Control-Message Protocol
IP: Internet Protocol
MAC: media-access control
NDDS: network data-delivery service
NFS: network-file system
RTPS: real-time publish-subscribe
TCP: Transmission-Control Protocol
UDP: User Datagram Protocol

A NETWORK-PROTOCOL PRIMER

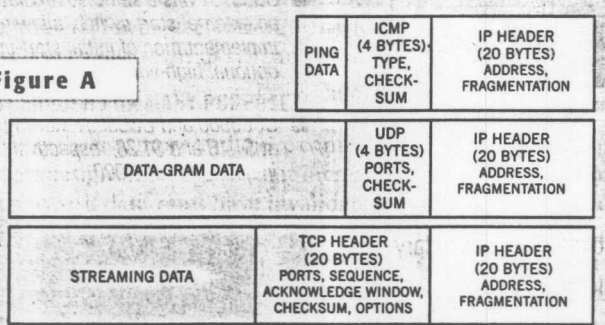
By Stan Schneider, PhD, President, Real-Time Innovations Inc

Everyone knows that the "standard" Internet protocol is TCP/IP. However, many don't know that the network stack commonly called "TCP/IP" also includes many other protocols, including IP, TCP, UDP, and ICMP. The entire stack runs on some hardware medium, such as Ethernet.

Ethernet is the most popular physical layer for high-speed networks. Ethernet sends information in approximately 1500-byte, individually routed "frames" to hardware MAC addresses. These addresses have a format such as 00:23:80:33:01:44. They specify a hardware-interface card and work only on one network. Ethernet frames are sent with no guarantee about the transmission, arrival, order, or much of anything else.

IP underlies nearly everything on the wire (**Figure A**). IP is a simple protocol that exists mostly to provide addressing (the ubiquitous "dot" addresses, such as 192.168.168.43). Routers use these addresses to send the information along the various hardware nets to the right destination. IP packet lengths may be as great as 64 kbytes. The IP layer fragments the messages into underlying transport packets (for example, Ethernet frames) and then re-integrates them on the receiving side. IP transmissions are not guaranteed; they may arrive out of order or not at all.

Figure A



ICMP ("ping," for example) is a simple protocol. UDP is a thin layer on top of IP that provides multiport addressing and a checksum. TCP is a more complex protocol that provides reliability and streaming data.

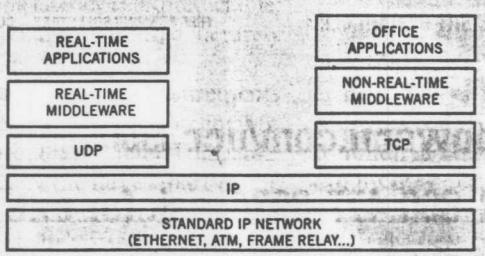


Figure B

Most network protocols are built on IP. UDP offers more timing control than TCP and is often better for real-time data distribution. TCP offers more reliable connections than UDP and is often better for enterprise networks. In both cases, middleware provides the application software with more services and an easier interface than do the lower level protocols.

ICMP is a simple protocol that sends control information between nodes. The best-known ICMP packets are "ping" messages, sent to determine whether a remote node is alive. Other ICMP packets control routing, fault isolation, and congestion control. ICMP messages are sent inside IP packets.

TCP also sends information inside IP packets. TCP provides a reliable, connection-oriented, streaming connection between two nodes. In other words, TCP breaks a stream of information into sizes that fit into IP packets, sends the packets to the other node, and then reassembles the packets into the same message that was sent. If this article were sent over a TCP connection, every character would arrive at the destination in the order in which it was sent. If an IP packet doesn't arrive, TCP asks the sending node to resend the packet and then delivers the stream intact. As a result, the receiving application doesn't have to worry about losing data; it sees an uninterrupted virtual "data stream."

UNDERLYING PROTOCOL

TCP obviously performs useful functions. It underlies most of the familiar higher level network protocols, such as HTTP (Web service), CORBA, and DCOM. However, TCP's advantages come at a cost. To provide a reliable connection, both nodes must maintain status information, such as what has been sent. As in a telephone network, it's difficult to maintain many of these connections to other nodes at the same time. Also, although the reliable protocol is nice when you want data no matter how long it takes to get it, the protocol does remove control over delivery timing. That loss of timing control can be a problem for real-time systems.

UDP is a thin layer around IP. It provides connectionless "datagram" packets with no guarantee of delivery. Using UDP is similar to mailing a letter: You don't really know whether it reaches the address you specified. UDP does, however, allow you to specify a receiving address, and it ensures that the information arrives uncorrupted if it arrives at all. If this article were sent as a series of UDP packets, pieces of it might not arrive or might arrive out of order, but whatever did arrive would be correct.

UDP is every bit as much of a part of the standard TCP/IP stack as is TCP (**Figure B**). Many common services are based on UDP, including the Windows NetBIOS system, the Unix NFS, and the DNS that resolves "www.rti.com" into an IP address.

Because there are no connections, UDP need not maintain a record of the state of the sending or receiving nodes. UDP can deliver the same packet to many addresses either on a local network (broadcast) or to multiple remote nodes (multicast). Therefore, UDP scales well to large one-to-many network communications. Because it lets the application control retried transmissions, UDP is often superior to TCP for real-time data distribution. For high-performance communications, protocols such as RealAudio and RealVideo (www.real.com), and real-time industrial middleware, such as NDDS, take advantage of this feature. It is obviously the job—and privilege—of these higher level protocols to control reliability and delivery timing.

time-control systems to model the systems and, before deployment, evaluate them more effectively than would otherwise be possible (see **sidebar** "Using the Web to make the most of mathematical models").

Several companies offer small Web servers that control-system designers can embed within higher level products. A company that offers an extensive array of hardware and software products, including specialized ICs and development tools, is NetSilicon.

Meanwhile, a brand-new company, Ipsil, has devised what it calls the smallest possible TCP/IP stack. Ipsil's initial product is small and inexpensive enough to make it a serious candidate for embedding within low-cost sensors and trans-

USING THE WEB TO MAKE THE MOST OF MATHEMATICAL MODELS

At least two companies are in the business of helping engineers and scientists use the Internet to distribute and use mathematical models and simulations. These models and simulations can represent the behavior of any system or process in any discipline—mechanical, electrical, electronic, chemical, biological, financial, and, no doubt, others. Modeling of real-time control systems is a prime example.

Historically, a large obstacle to the distribution of such models has been the developers' insistence on retaining control of their intellectual property. That is, the developers insist on retaining control of the equations and algorithms that underlie the behavior of the model. Web technology offers a means of keeping models' proprietary parts locked behind a firewall yet allows users to input data and observe results. Web technology also allows control of who can use the models and can enable intellectual-property owners to charge for the use of their property.

Users of MatLab and other scientific and technical computing tools from The MathWorks can distribute models in several ways, one of which, the MatLab Web Server, depends strongly on Web technology.

Of The MathWorks' approaches, the most straightforward and one of the least costly uses the MatLab Compiler. With the Compiler, a model's owner can distribute compiled MatLab code by any conventional means, such as mailing disks, attaching files to e-mail, or providing downloads. Unfortunately, this approach is not particularly secure. For most model owners and users, C code is the most convenient form of output from the Compiler because it relieves the model user of the need to have MatLab to run the code. But C programmers can reverse-engineer the code—albeit not always completely.

MatLab Run-time Server software is a somewhat more expensive approach that provides greater security. The output of the Run-time Server is pseudo-code (P-code), which the model owner can distribute by the same means as those for C code. P-code is much less transparent, however. The Run-time Server can bundle an interpreter with the P-code, enabling the code to run on machines on which MatLab is not installed. In many cases, though, execution is slower than that of the equivalent C program.

THE MATLAB WEB SERVER

For many MathWorks customers, MatLab Web Server software is the best choice for distributing models. The Web Server approach is not only economical, but also provides the greatest security because the code need never leave the intellectual-property owner's server hardware. Moreover, the Web Server supports not just MatLab but also The MathWorks' Simulink product, whose stock in trade is, as its name implies, simulation.

When you use the MatLab Web Server to distribute a model, you create an HTML interface through which users supply data to the model. The model's output is HTML as well—often HTML tables. Because spreadsheets, such as Excel, accept HTML tables as inputs, you can easily perform further analysis on the model or simulation output.

In one respect, however, the MatLab Web Server is at a disadvantage with respect to the MatLab Compiler and Run-time Server. Web Server models produce static graphical output. For example, if you want to see how a 3-D graph looks from a different viewpoint, you can't rotate the picture as you can with graphics produced by code that the other two tools generate. Instead, you must specify a new azimuth and elevation for the viewpoint, and you can do that only if the model accepts those parameters as input.

INNOVATION CHAIN

Start-up Innovation Chain's business is complementary to The MathWorks' model-distribution activities. Innovation Chain focuses on the infrastructure for Web-based model and simulation distribution rather than on the applications that generate or run the models. For example, if the intellectual-property owner prefers to provide free access to its models, Innovation Chain can provide its infrastructure tools to enable such access. Publishers of periodicals such as *NASA Tech Briefs* distribute models via Innovation Chain's servers and promote those models through articles in the publication. Users access the models via links from the publisher's Web site (for NASA Tech briefs, the URL is www.innovationchain.com/first.asp?GN=111).

Some property owners may want only their employees, suppliers, or customers to be able to access certain models. In such cases, it can be appropriate to use a customized version of the Innovation Chain infrastructure to distribute the models from the owner's network

Innovation Chain's approach allows intellectual-property owners to use their own executables or to use any vendor's software to develop the models and simulations. This agnostic approach to the underlying software is possible because of the use of Excel spreadsheets as an interface and control panel. The interface sends user-supplied data to the remote models, launches them, and returns the simulation results when execution is complete but doesn't allow users to access the model's underlying algorithms. A proprietary plug-in enables Excel to communicate with the remote server and to handle complex data types, such as meshes and fields. The free plug-in is available for downloading from Innovation Chain's Web site.

Additional features of Innovation Chain's approach include providing different classes of users with different levels of access to models. For example, site administrators might permit only those users who contract for a high level of access to solve problems. Other users might be referred instead to prepackaged "best-match" solutions of similar problems. Furthermore, Innovation Chain's tools can provide intellectual-property owners with feedback on the popularity of various aspects of each model or simulation.

ducers. The product is an IC that contains the stack, a µP core, I/O ports, a low-speed network port, and both RAM and ROM. The device, which contains approximately 5000 gates in an eight-pin surface-mount package, will cost less than $1 in production quantities.

Ipsil also plans to offer the product as a standard cell that designers of higher level products can integrate into more complex ICs. For such designers, Ipsil offers a $100 development kit that includes the IC as well as development notes and application examples. A version of the IC that incorporates a 10BaseT Ethernet interface would contain only about 20,000 gates.

Nevertheless, you shouldn't get the idea that Ipsil wants to see thousands of sensors in a manufacturing plant all publishing their own Web pages to the company intranet. Box-level products, such as Capital Equipment's WebDaq, a data-acquisition system, publish Web pages that contain data acquired from multiple sensors. This approach eliminates the need for specialized client software. All you need to view a WebDaq page is a networked computer equipped with a standard Web browser.

TRILLIONS OF DEVICES

Although it has nothing against this approach, Ipsil has its eye on lower level products that sensor manufacturers produce by the thousands and manufacturers of information appliances may produce by the millions. Indeed, Ipsil believes that the total market for its minimalist technology could amount to trillions of units. Vendors of box-level data-acquisition products often sell single units and only occasionally take individual orders for quantities of more than 100 units.

Ipsil's view is that publishing the data to an intranet should be the responsibility of a higher level server that aggregates the data from many sensors. Efficient use of network resources is paramount in such architectures, so despite their compactness, simplicity, and low cost, Ipsil's devices can incorporate RTPS middleware as an embedded feature.

Indeed, embedding such a software layer within devices such as sensors makes enormous sense. Even though networks have taken tremendous strides over the last decade in increasing speed and reducing cost, interconnection is still the most significant part of sensor-based systems' installed cost. The cost of functions that silicon chips can incorporate has declined much faster than the cost of wired—or wireless—interconnects. Using sensors that employ embedded intelligence to limit the amount of transmitted data is far more cost-effective than increasing the network bandwidth, collecting more data, and trying to determine what to make of the data as it fills—and then overflows from—massive networked storage devices.□

You can reach Senior Technical Editor Dan Strassberg at 1-617-558-4205, fax 1-617-558-4470, e-mail ednstrassberg@cahners.com.

REFERENCE

1. Schneider, Stan, "Making Ethernet work in real time," *Sensors*, November 2000, pg 32.

FOR MORE INFORMATION...

For more information on products such as those discussed in this article, go to our information-request page at www.rscahners.ims.ca/ednmag/. When you contact any of the following manufacturers directly, please let them know you read about their products in *EDN*.

Avantcom Network
1-408-436-2171
www.avantcom.net
Enter No. 301

Capital Equipment Corp
1-978-663-2002
www.cec488.com
Enter No. 302

Datasweep Inc
1-408-350-7300
www.datasweep.com
Enter No. 303

GE Cisco Industrial Networks
1-800-327-8262
www.gecisco.com
Enter No. 304

Innovation Chain
1-781-890-5091
www.innovationchain.com
Enter No. 305

Intelligent Instrumentation Inc
1-800-685-9911
www.instrument.com
Enter No. 306

Ipsil
1-617-441-0690
www.ipsil.com
Enter No. 307

Jumptec–Adastra Systems Corp
1-510-732-6900
www.jumptec.de
Enter No. 308

Labtech
1-978-470-0099
www.labtech.com
Enter No. 309

The MathWorks
1-508-647-7000
www.mathworks.com
Enter No. 310

NetSilicon Inc
1-800-243-2333, 1-781-647-1234
www.netsilicon.com
Enter No. 311

Real-Time Innovations
1-408-734-4200
www.rti.com
Enter No. 312

Rockwell Automation
1-414-382-2000
www.automation.rockwell.com
Enter No. 313

Schneider Automation Inc
1-978-794-0800
www.schneiderautomation.com
Enter No. 314

Syntricity
1-858-552-4485
www.syntricity.com
Enter No. 315

Additional resources for information on Web-based industrial automation:

The Industrial Automation Open Networking Alliance
www.IAONA.com

The Industrial Ethernet Book (a Web-based "book")
http://ethernet.industrial-networking.com

SUPER INFO NUMBER
For more information on the products available from all of the vendors listed in this box, enter No. 316 at www.rscahners.ims.ca/ednmag/.